ARL-TN-0789 • Sep 2016

US Army Research Laboratory

3-D Synthetic Microstructure Generation with Ellipsoid Particles

by Mark A Tschopp

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-0789 ● SEP 2016

US Army Research Laboratory

3-D Synthetic Microstructure Generation with Ellipsoid Particles

by Mark A Tschopp
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2016	Technical Note	January 2016–June 2016

4. TITLE AND SUBTITLE
3-D Synthetic Microstructure Generation with Ellipsoid Particles

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Mark A Tschopp

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-WMM-F
Aberdeen Proving Ground, MD 21005-5069

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-0789

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Synthetic 2-phase microstructures are often used as a surrogate for real microstructures based on experimental microstructure statistics or as a way to test how different microstructure features affect properties of materials. These can be generated through a number of different techniques. In this technical note, a random sequential adsorption algorithm implemented in MATLAB is used to generate 3-dimensional synthetic microstructures composed of packing of ellipses within a voxelized microstructure. The MATLAB scripts are attached as appendices for future development and application.

15. SUBJECT TERMS
microstructure, MATLAB, synthetic, random sequential adsorption, ellipse

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mark A Tschopp
Unclassified	Unclassified	Unclassified	UU	32	19b. TELEPHONE NUMBER (Include area code) 410-306-0855

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

Synthetic 2-phase microstructures are often used as a surrogate for real microstructures based on experimental microstructure statistics or as a way to test how different microstructure features affect properties of materials. These can be generated through processes such as the random sequential algorithm described within or other techniques, some of which are outlined in the book on heterogeneous microstructures by Torquato.[1] Random sequential adsorption (RSA) has been applied to understand microstructure and phenomena in a number of different systems and applications, including far-from-equilibrium processes,[2] car parking and protein adsorption,[3] colloids,[4] dimer adsorption,[5] fiber composites,[6] and even disks,[7] rectangles[8,9] and n-star objects.[10]

In this technical note, a random sequential adsorption[11,12] algorithm implemented in MATLAB[13] is used to generate three-dimensional (3-D) synthetic microstructures composed of packing of ellipses within a voxelized microstructure. The MATLAB scripts are attached as appendices for future development and application. This work is an extension of a previous two-dimensional (2-D) synthetic microstructure builder that incorporates 2-D ellipses, aspect ratio, area fraction, size (polydispersivity), orientation, and size/orientation distributions.[14]

2. Algorithm/Code Description

The algorithm used to generate the synthetic microstructures is an RSA algorithm, which is typically used to represent particle deposition on surfaces. For instance, there are multiple stages in particle deposition: 1) initially, the surface is clean and free of particles, 2) then particles begin to spontaneously attach to the surface at small concentrations, and 3) the deposition progressively slows down due to deposited particles blocking subsequent deposition of other particles. This process is typically studied by the RSA model (for more information, see Torquato[1]), which occurs via the following steps:

- A spherical/ellipsoidal particle is randomly placed on the surface. Once placed, this particle's position is fixed and it cannot move.
- A trial spherical/ellipsoidal particle is randomly placed. If this particle overlaps previously placed particle(s), then this attempt is rejected. Otherwise, the placement is accepted.

- The model generally proceeds by adding until a predetermined area/volume fraction of particles is reached or the jamming limit is reached. The jamming limit or saturation of the surface occurs when it is no longer possible to place a particle within the holes of the other deposited particles.

3. Implementation and Usage

This is implemented through the MATLAB scripts in Appendix A, Appendix B, and Appendix C by using 3-D matrices, where the background is 0 and the particle is 1. For the 3-D ellipses, it is possible to alter the 3 orthogonal axes of the ellipse (i.e., spheres, plates, or needles), the orientations of the ellipses, and the volume fraction. In both cases, it is relatively straightforward to include distributions of ellipse shapes, ellipse orientations, or ellipse sizes. Appendix A is the main code, Appendix B is the optimized ellipse builder function, and Appendix C includes a few functions called within the script.

The attached scripts have been tested on MATLAB R2014 and R2015 on a Windows operating system. The code can be executed by following these steps:

- Download the various scripts into the same directory:
  - generate_synthetic_structure_ellipse_3D_anisotropic.m (download)
  - image_ellipse_3D_fast.m (download)
  - image_view.m (download)
  - deleteEmptyExcelSheets.m (download)
- Open the script “generate_synthetic_structure_ellipse_3D_anisotropic.m” in MATLAB
- Type “generate_synthetic_structure_ellipse_3D_anisotropic” at the command prompt to run.

Different 3-D microstructures can be obtained by adjusting the initialization parameters volume fraction (Vf_max), ellipse axis lengths (a,b,c), and image size (e.g., the “I=false(256,256,256);” statement). The approximate number of particles are calculated to give an approximate indication of the time that may be required. For

higher-volume fractions, the code will have a progressively harder time placing particles as less matrix space is available. The code supplied can be easily extended to incorporate size distributions, aspect ratio distributions, or orientation distributions, as has been done in previous work.[14]

4. Examples

This code was developed to examine the structure-property relationship based on the influence of particle aspect ratio, area fraction, packing/clustering, and particle orientation. For instance, prior work has examined metrics for characterizing length scales in different composite microstructures.[15,16] The voxelized microstructures can be output as images or meshes for subsequent calculations. Also, the parameters used to generate the synthetic microstructures can be output for building within another software package (i.e., box dimensions in x,y,z coordinates, center of ellipse in x,y,z coordinates, orthogonal major/minor axis lengths of the ellipse [aspect ratio], and the orientation of the ellipse in Euler angles [ψ_1,ψ_2,ϕ]). For instance, Fig. 1 shows an example of a subset of 2-D slices from a 3-D synthetic microstructure with over 13,000 particles placed at a volume fraction of 30%, where the aspect ratio is increased from spherical (1:1:1) to more needle-like (4:1:1). Since all ellipses in these microstructures are identical, Fig. 1c shows how difficult it is to deduce the 3-D microstructure from 2-D microstructure slices (i.e., this microstructure appears to have particles with a distribution of aspect ratios). Figure 2 illustrates the full 2-D slice for this microstructure to illustrate this more clearly. Notice that the periodic boundaries are somewhat visible in this image for some of the particles where the major axis of the ellipse is perpendicular to the viewing direction (i.e., they appear more elliptical in the cross-section image). Figure 3 illustrates the full 2-D slice for a microstructure with randomly oriented ellipses that have an 8:8:1 aspect ratio (i.e., plates). Again, the 2-D image appears to contain a mix of particle sizes and shapes when in reality, these particles are all the same size.

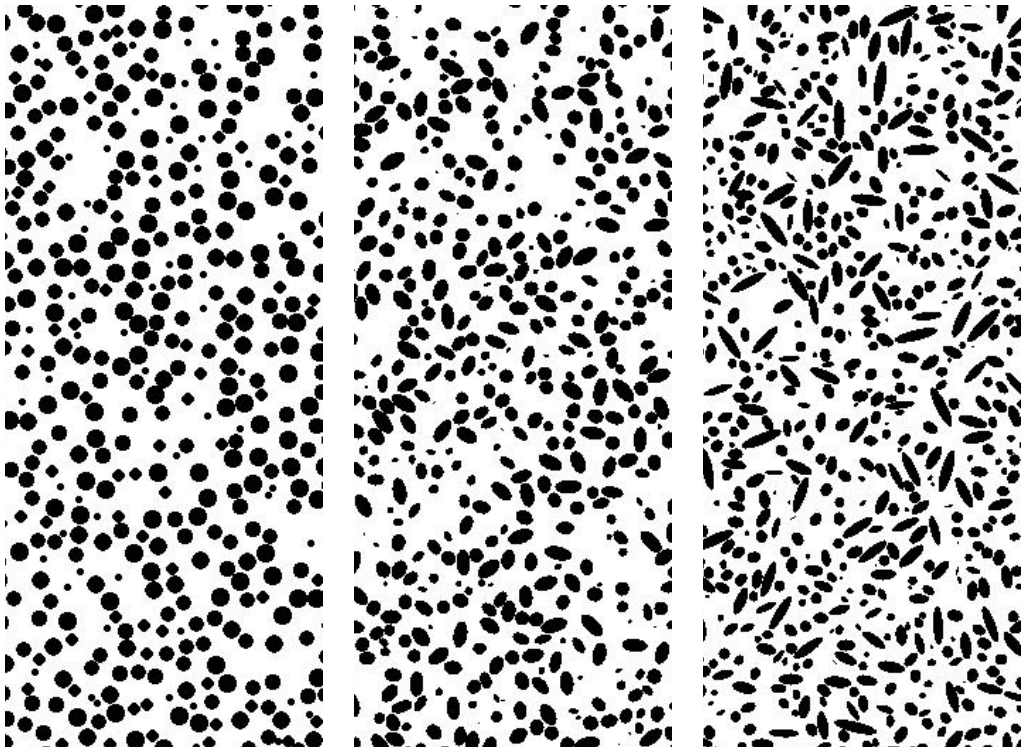

Fig. 1 2-D images of 3-D synthetic microstructures with uniform size circles/ellipses that are randomly oriented with aspect ratios of a) 1:1:1, b) 2:1:1, and c) 4:1:1. The volume fraction in this microstructure is 30% for all microstructures. For the volume computed, there were over 13,000 particles placed for each of the cases.

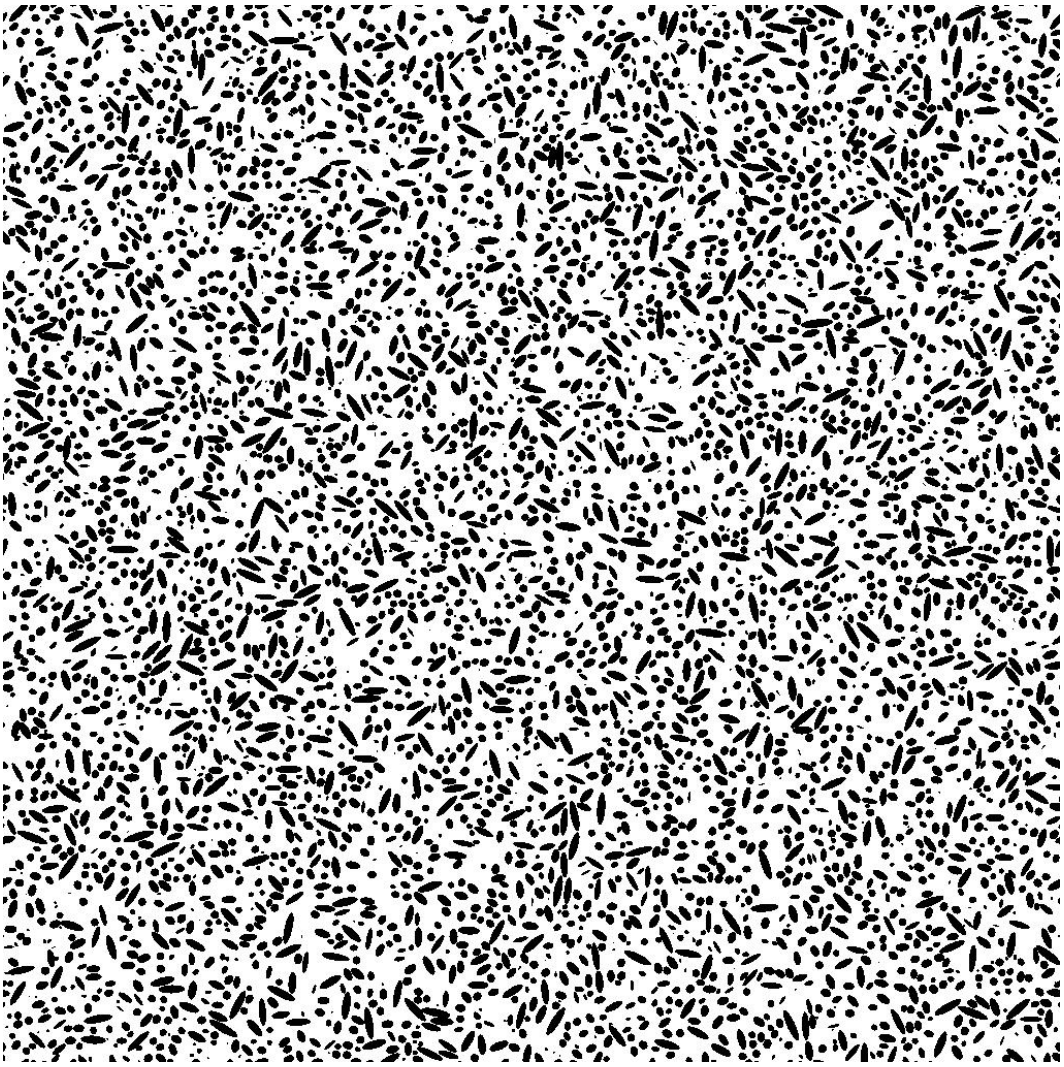

Fig. 2 2-D images of 3-D periodic synthetic microstructure with uniform ellipses of 4:1:1 aspect ratio that are randomly oriented. The volume fraction in this microstructure is 30%. For the volume computed, there were 14,133 particles placed.

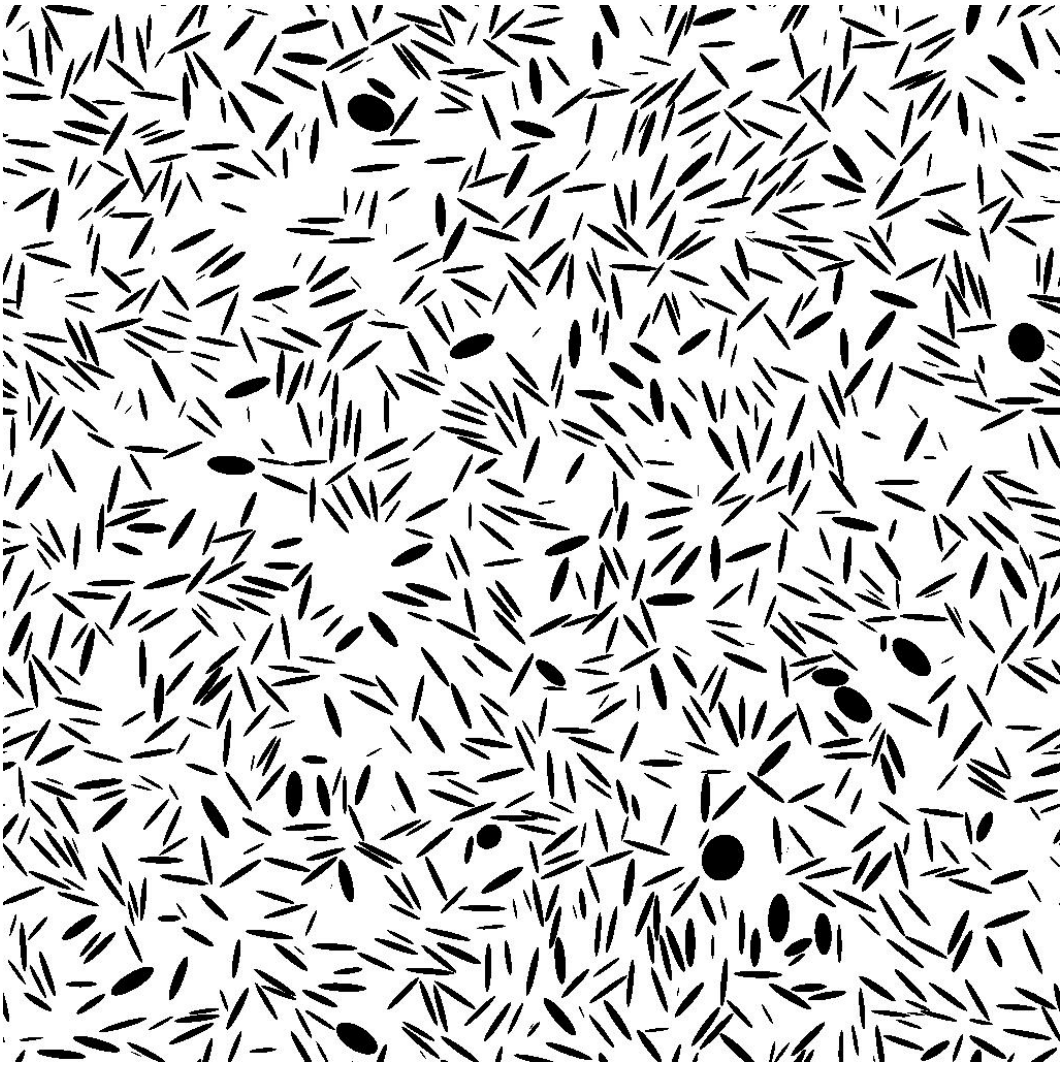

Fig. 3 2-D images of 3-D periodic synthetic microstructure with uniform ellipses of 8:8:1 aspect ratio (i.e., plates) that are randomly oriented. The volume fraction in this microstructure is 19%. For the volume computed, there were 1,326 particles placed.

5. Conclusion

Synthetic 2-phase microstructures are often used as a surrogate for real microstructures based on experimental microstructure statistics or as a way to test how different microstructure features affect properties of materials. In this technical note, an RSA algorithm was implemented in MATLAB to generate 3-D synthetic microstructures composed of packing of ellipses within a voxelized microstructure. The MATLAB scripts are attached as appendices for future application or development.

6. References

1. Torquato S. Random heterogeneous materials: microstructures and macroscopic properties. New York (NY): Springer; 2002.

2. Evans JW. Random and cooperative sequential adsorption. Rev Modern Phys. 1993;65:1281.

3. Talbot J, Tarjus G, Van Tassel PR, Viot P. From car parking to protein adsorption: an overview of sequential adsorption processes. Colloids Surf A: Physiochem Eng Aspects. 2000;165:287–324.

4. Adamczyk Z. Modeling adsorption of colloids and proteins. Curr Opin Colloid & Interf Sci. 2012;17(3):173–186.

5. Cieśla M, Barbasz J. Modelling of interacting dimer adsorption. Surf Sci. 2013;612:24–30.

6. Lu Z, Yuan Z, Liu Q. 3D numerical simulation for the elastic properties of random fiber composites with a wide range of fiber aspect ratios. Comp Mater Sci. 2014;90:123–129.

7. Shelke PB, Limaye AV. Dynamics of random sequential adsorption (RSA) of linear chains consisting of n circular discs – role of aspect ratio and departure from convexity. Surf Sci. 2015;637–638:1–4.

8. Vigil RD, Ziff RM. Random sequential adsorption of unoriented rectangles onto a plane. J Chem Phys. 1989;91:2599–2602.

9. Dickman R, Wang JS, Jensen I. Random sequential adsorption: series and virial expansions. J Chem Phys. 1991;94:8252–8257.

10. Shelke PB. Random sequential adsorption of n-star objects. Surf Sci. 2016;644:34–40.

11. Widom B. Random sequential addition of hard spheres to a volume. J Chem Phys. 1966;44:3888.

12. Feder J. Random sequential adsorption. J Theoret Biol. 1980;87(2):237–254.

13. MATLAB. Release 2015. Natick (MA): The MathWorks, Inc.; 2015.

14. Tschopp MA. Synthetic microstructure generator. Version 1.2. Natick (MA): MathWorks; 2009 Sep 23 (updated 2009 Nov 10) [accessed 2016 Aug 25]. http://www.mathworks.com/matlabcentral/fileexchange/25389-synthetic-microstructure-generator/.

15. Tschopp MA, Wilks GB, Spowart JE. Multi-scale characterization of orthotropic microstructures. Modell Simul Mater Sci Eng. 2008;16(6):065009.

16. Wilks GB, Tschopp MA, Spowart JE. Multi-scale characterization of inhomogeneous morphologically textured microstructures. Mater Sci Eng. 2010;527(4):883–889.

INTENTIONALLY LEFT BLANK.

Appendix A. MATLAB Scripts: Main Code

This appendix appears in its original form, without editorial change.

```
%% Generate Synthetic 3D Structure with Ellipses
% Mark Tschopp
% 2016
% Random Sequential Adsorption algorithm that varies
% ellipse volume fraction, size, aspect ratio, and
% orientation

clear all; clc;

% Initialize variables used in generation of 3D microstructure
tic
Vf = 0; Vf_max = 0.05;
a = 12; b = 6; c = 6;
nparticles = 0;
I = false(256,256,256);
image_size = size(I);

% Calculate approximate Vf of particle and approximate number of
% particles to generate the specified volume fraction.  This
% will impact the time that it takes to render the 3D volume.

% Assign random orientation
psi1 = 2*pi*rand; psi2 = 2*pi*rand; phi = acos(rand);
I_ellipse = image_ellipse_3D_fast(a,b,c,psi1,psi2,phi);
Vf_particle = sum(I_ellipse(:)) / numel(I);
approx_nparticles = Vf_max/Vf_particle;
disp(['Approximate # of particles: ' num2str(approx_nparticles)]);

% Main loop for generating microstructure
h = waitbar(0,'Ellipsoid placement');
while Vf < Vf_max

    psi1 = 2*pi*rand;
    psi2 = 2*pi*rand;
    phi = acos(rand);

    I_ellipse = image_ellipse_3D_fast(a,b,c,psi1,psi2,phi);

    x0 = ceil(rand*image_size(1));
    y0 = ceil(rand*image_size(2));
    z0 = ceil(rand*image_size(3));

    diam = size(I_ellipse,1);
```

```
% Merge circle matrix with synthetic microstructure image
nlo = x0 - floor(diam/2); nhi = x0 + ceil(diam/2)-1;
ix = mod(nlo:nhi,image_size(1));ix(ix==0)=image_size(1);
nlo = y0 - floor(diam/2); nhi = y0 + ceil(diam/2)-1;
iy = mod(nlo:nhi,image_size(2));iy(iy==0)=image_size(2);
nlo = z0 - floor(diam/2); nhi = z0 + ceil(diam/2)-1;
iz = mod(nlo:nhi,image_size(3));iz(iz==0)=image_size(3);

Itest = logical(I(ix,iy,iz));
if sum(Itest(I_ellipse)) == 0
    Itest(I_ellipse) = 1;
    I(ix,iy,iz) = Itest;
    accept = 1;
else
    accept = 0;
    iter = 0;
    while accept == 0 && iter <= 10
        iter = iter + 1;
        x0 = ceil(rand*image_size(1));
        y0 = ceil(rand*image_size(2));
        z0 = ceil(rand*image_size(3));
        diam = size(I_ellipse,1);

        % Merge circle matrix with synthetic microstructure
        nlo = x0 - floor(diam/2); nhi = x0 + ceil(diam/2)-1;
        ix = mod(nlo:nhi,image_size(1));
        ix(ix==0)=image_size(1);
        nlo = y0 - floor(diam/2); nhi = y0 + ceil(diam/2)-1;
        iy = mod(nlo:nhi,image_size(2));
        iy(iy==0)=image_size(2);
        nlo = z0 - floor(diam/2); nhi = z0 + ceil(diam/2)-1;
        iz = mod(nlo:nhi,image_size(3));
        iz(iz==0)=image_size(3);

        Itest = logical(I(ix,iy,iz));
        if sum(Itest(I_ellipse)) == 0
            Itest(I_ellipse) = 1;
            I(ix,iy,iz) = Itest;
            accept = 1;
        end
    end
end
```

```
    if accept;
        nparticles = nparticles + 1;
        Vf_particle = sum(I_ellipse(:)) / numel(I);
        Vf = Vf + Vf_particle;
        ellipse(nparticles,1:10) = ...
              [x0 y0 z0 a b c psi1 psi2 phi Vf_particle];
    end

    waitbar(Vf/Vf_max)
end
close(h)
disp(['3D Digital Slices (sec): ' num2str(toc)]);
disp(['Number of particles: ' num2str(nparticles)]);

%% Write ellipse parameters to Excel
% Store all ellipse parameters in Excel so that the 3D structure
% can be generated without worrying about overlap.

image_save_flag = 1;
if image_save_flag

    string_path = pwd;
    Excel_fileName = sprintf('%s\\3D_ellipses.xlsx',string_path);
    sheet = sprintf('ellipse_%dvf_%da_%db_%dc_%05d',...
    round(100*Vf),a,b,c,nparticles);
    warning off MATLAB:xlswrite:AddSheet;
    ColHeaders = {'x0','y0','z0','a','b','c','psi1','psi1','phi'};
    xlswrite(Excel_fileName, ColHeaders, sheet, 'A1');
    xlswrite(Excel_fileName, ellipse, sheet, 'A2');
    deleteEmptyExcelSheets(Excel_fileName);

end

%% View slices of a 3D section
% This routine views 2D orthogonal planes in the 3D structure as a
% function of depth

j=1;
close all;
figure;
while j
    for i = 1:image_size(3)
        J = 1-I(:,:,i);
```

```
        image_view(J);
        title(['Slice ' num2str(i)])
        pause(0.05)
    end
end

% Hit control-C to exit

%% Save individual slices as jpegs for animated gifs
% For use with ImageJ - First, open ImageJ and select Plugins -> List
% Opener, then select the *.txt file that contains all the image
% paths. Once the images are loaded in ImageJ, then convert them to
% Slices using the Image -> Stacks -> Image to Stacks command.  Now
% the stack of images can be saved as an animated *.gif format for
% insertion into Powerpoint.  The Volume viewer and VolumeJ renderers
% in ImageJ also accept stacks of images.

tic
close all; figure;
sheetname = sprintf('ellipse_%dvf_%da_%db_%dc_%05d',...
    round(100*Vf),a,b,c,nparticles);
if ~isdir(sheetname), mkdir(sheetname); end
copyfile('image_view.m',sheetname)
cd(sheetname)
fid = fopen([sheetname '.txt'],'w');
for i = 1:image_size(3)
    K = reshape(1-I(:,:,i),image_size(1),image_size(2));
    J = K;
    image_view(J);
    pause(0.05)
    image_filename = [sheetname, sprintf('_%03d.jpg',i)];
    fprintf(fid,'%s \n',image_filename);
    if image_save_flag; imwrite(J,image_filename,'jpg'); end
end
fclose(fid);
delete('image_view.m')
cd ..

% Save to *.mat file as well
save([sheetname '.mat'],'I')

disp(['Image viewing and saving (sec): ' num2str(toc)]);
```

```
% *.mat file is useful for subsequent operations in MATLAB or writing
% files, *.jpg is useful for visualization, *.tif binaries are a useful
% if lossless compression is desired, *.txt contains a list of images,
% *.xls contains spreadsheet with all ellipse parameters, if desired
```

Appendix B. MATLAB Function: Create Ellipse

This appendix appears in its original form, without editorial change.

```
function y = image_ellipse_3D_fast(a, b, c, psi1, psi2, phi)

% Mark Tschopp
% 2016
% This subroutine generates a 3D ellipse in a matrix (y) using the
% orthogonal distances in the ellipse (a,b,c) and the Euler angles
% (psi1,psi2,phi), which define the rotation of the ellipse.

%{
% These parameters can be used for debug purposes.
a = 40;
b = 10;
c = 10;
a = ceil(rand*40);
b = ceil(rand*40);
c = ceil(rand*40);
psi1 = 2*pi*rand;
psi2 = 2*pi*rand;
phi = acos(rand);
%}

% First, identify what the size of the matrix must be to contain the
% ellipse and also what the central symmetry plane will be (ic).

if a > b; diameter = 2*a; else diameter = 2*b; end
if 2*c > diameter; diameter = 2*c; end
if mod(diameter,round(diameter))~= 0; diameter = ceil(diameter); end
radius = diameter/2;
if mod(radius,round(radius))~= 0;
    diameter = diameter+1; radius = diameter/2;
end
dist(1:diameter+1,1:diameter+1,1:diameter+1) = 2;
ic = 1 + radius;

% M is the rotation matrix for the ellipse, based on the Euler angles
% psi1, psi2, and phi.  Different rotation matrices and Euler angle
% conventions can be used, if necessary.

M = zeros(3);
M(1,1) = cos(psi1)*cos(psi2)-sin(psi1)*sin(psi2)*cos(phi);
M(2,1) = -cos(psi1)*sin(psi2)-sin(psi1)*cos(psi2)*cos(phi);
M(3,1) = sin(psi1)*sin(phi);
M(1,2) = sin(psi1)*cos(psi2)+cos(psi1)*sin(psi2)*cos(phi);
```

```
M(2,2) = -sin(psi1)*sin(psi2)+cos(psi1)*cos(psi2)*cos(phi);
M(3,2) = -cos(psi1)*sin(phi);
M(1,3) = sin(psi2)*sin(phi);
M(2,3) = cos(psi2)*sin(phi);
M(3,3) = cos(phi);

% This step creates a matrix with the distances to the ellipse
% centroid (ic, ic, ic).  This has been optimized to minimize the
% time to generate the 3D ellipse.  The changes resulted in a
% 94% decrease in the cpu time required to generate the ellipse.

k = ic-1;
i0 = 1; i1 = diameter + 1;
j0 = 1; j1 = diameter + 1;
while k < diameter+1
    k = k + 1;

    % This loops over all the pixels in the first plane to find
    % the pixels belonging to the ellipse

    for i = i0:i1
        for j = j0:j1
            a1 = [i-ic, j-ic, k-ic];
            a2 = [a^2, b^2, c^2];
            c1 = a1*M;
            c2 = c1.^2./a2;
            dist(i,j,k) = sum(c2);
        end
    end

    % If there are no pixels belonging to the ellipse on this plane,
    % then go ahead and exit out of the loop by setting k to the
    % final plane

    if sum(sum(dist(:,:,k)<=1))==0
        k = diameter + 1;
    end

    % If this is not the first or last plane, then the program
    % is smarter about which pixels it checks for ellipse pixels in

    if k ~= ic && k ~= diameter + 1
        d = dist(:,:,k-1) <= 1;
```

```
        e = dist(:,:,k) <= 1;
        d1 = diff(sum(d,1)~=0);  dmin = find(d1 == 1);
        dmax = find(d1 == -1);
        if size(dmin,2) > 1; dmin = dmin(1); dmax = dmax(2); end
        e1 = diff(sum(e,1)~=0);  emin = find(e1 == 1);
        emax = find(e1 == -1);
        if size(emin,2) > 1; emin = emin(1); emax = emax(2); end
        if dmin - emin < 0;
            j0 = emin - 1;
            j1 = j0 + 3 + (dmax-dmin);
            if j1 > diameter + 1; j1 = diameter + 1; end
        end
        if dmax - emax > 0;
            j1 = emax + 1;
            j0 = j1 - 3 - (dmax-dmin);
            if j0 < 1; j0 = 1; end
        end

        d1 = diff(sum(d,2)~=0);  dmin = find(d1 == 1);
        dmax = find(d1 == -1);
        if size(dmin,1) > 1; dmin = dmin(1); dmax = dmax(2); end
        e1 = diff(sum(e,2)~=0);  emin = find(e1 == 1);
        emax = find(e1 == -1);
        if size(emin,1) > 1; emin = emin(1); emax = emax(2); end
        if dmin - emin < 0;
            i0 = emin - 1;
            i1 = i0 + 3 + (dmax-dmin);
            if i1 > diameter + 1; i1 = diameter + 1; end
        end
        if dmax - emax > 0;
            i1 = emax + 1;
            i0 = i1 - 3 - (dmax-dmin);
            if i0 < 1; i0 = 1; end
        end
    end
end

%% Generate whole ellipse
% This section uses symmetry of the ellipse about the center plane
% to generate the entire ellipse in matrix y.  The defined distance
% matrix in the previous section has a threshold of 1.  Distances
% less than or equal to 1 belong to the ellipse; all other distances
% do not.
```

```
y = dist <= 1;
x = y(1:diameter+1,1:diameter+1,ic + 1:diameter+1);
x1(1:diameter+1,1:diameter+1,1:ic-1) = 0;
for i = 1:size(x,3)
    x1(:,:,size(x,3)-i+1) = flipud(fliplr(x(:,:,i)));
end
y(1:diameter+1,1:diameter+1,1:ic-1) = x1;

%% Show images
% For debug purposes, change the 'while 0' to 'while 1' for a stack
% of images through the ellipse.

while 0
    for i = 1:length(y)
        image_view(1-y(:,:,i));
        pause(0.15)
    end
end
```

INTENTIONALLY LEFT BLANK.

Appendix C. Other MATLAB Functions

This appendix appears in its original form, without editorial change.

```
function image_view(I)

% Mark Tschopp
% 2016
% Quick image view function that doesn't require
% the MATLAB Image Processing toolbox

imagesc(I);
colormap(gray);
axis off;
axis equal;
```

```
% created by: Quan Quach
% date: 11/6/07
% this function erases any empty sheets in an excel document

function deleteEmptyExcelSheets(fileName)

% the input fileName is the entire path of the file
% e.g., fileName = 'C:\Documents and Settings\matlab\myExcelFile.xls'

excelObj = actxserver('Excel.Application');
% opens up an excel object
excelWorkbook = excelObj.workbooks.Open(fileName);
worksheets = excelObj.sheets;
% total number of sheets in workbook
numSheets = worksheets.Count;

count=1;
for x=1:numSheets
    % stores the current number of sheets in the workbook
    % this number will change if sheets are deleted
    temp = worksheets.count;

    % if there's only one sheet left, we must leave it or else
    % there will be an error.
    if (temp == 1)
        break;
    end

    % this command will only delete the sheet if it is empty
    worksheets.Item(count).Delete;

    % if a sheet was not deleted, we move on to the next one
    % by incrementing the count variable
    if (temp == worksheets.count)
        count = count + 1;
    end
end
excelWorkbook.Save;
excelWorkbook.Close(false);
excelObj.Quit;
delete(excelObj);
```

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	RDRL WMM F M TSCHOPP